EXACT STATE COMPLEXITY OF THE SHIFTS OF THE FIBONACCI–THUE–MORSE SEQUENCE

KOREA SUPERINTELLIGENCE LABS

Abstract. Let $t(i)$ be the number of ones in the Zeckendorf representation of i, reduced modulo 2 (the Fibonacci–Thue–Morse sequence, OEIS A095076), and let $\mathrm{sc}(c)$ be the number of states of the smallest automaton with output that reads a Zeckendorf representation most-significant-digit first, reaches no state on an input containing two consecutive ones, and outputs $t(i+c)$ on every representation of i. Moradi, Rampersad and Shallit proved $\mathrm{sc}(c) = O(c)$ and asked for a matching lower bound (their Problem 1: "Prove the number of states in a minimal automaton generating $(t(i+c))_{i\geq 0}$ is $\Theta(c)$"). We determine $\mathrm{sc}(c)$ exactly for every $0 \leq c \leq 100$: the values run from $\mathrm{sc}(0) = 4$ to $\mathrm{sc}(100) = 568$, they are all even, they are not monotone, and $5c < \mathrm{sc}(c) \leq 7c$ for $2 \leq c \leq 100$. Each value is certified from below by a list of $\mathrm{sc}(c)$ pairwise inequivalent input words and from above by the minimised automaton of the Moradi–Rampersad–Shallit construction, whose transition rule — the Dekking morphism $A \to AD$, $B \to A$, $C \to CB$, $D \to C$ acting on windows of the pair sequence $(t(i), f(i))$, f the Fibonacci word — is proved correct for every shift and every window length. The case $c = 0$ shows that the four-state automaton for t given by Shallit is minimal. Nothing is claimed for general c: the linear lower bound of Problem 1 remains open, and we record the exact values to $c = 300$ and an empirical first-difference rule for the size of the construction as computations only. All theorems are machine-checked in Lean 4.

1. Introduction

Every non-negative integer i has a unique *Zeckendorf* (Fibonacci) representation $i = \sum_{k\geq 2} e_k F_k$ with digits $e_k \in \{0, 1\}$ and no two consecutive digits equal to 1, where $F_1 = F_2 = 1$ and $F_{k+2} = F_{k+1} + F_k$ [6, 5]. Reading the digits most significant first gives a binary word (i), and two sequences of digit statistics have received attention in combinatorics on words:

$$t(i) = \big(\text{number of ones in } (i)\big) \bmod 2, \qquad f(i) = \text{last digit of } (i).$$

The first is the *Fibonacci–Thue–Morse sequence* $t = 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, \ldots$ (OEIS A095076 [7]); the second is the Fibonacci word $f = 01001010\cdots$ [1, §2.2]. Both are *Fibonacci-automatic*: a finite automaton with output reading (i) most significant digit first computes them, with two states for f [1, §2.2] and four for t [4].

Moradi, Rampersad and Shallit [1] study how the number of states of such an automaton behaves under a shift of the sequence. Their Theorem 14 reads:

> Let $(t(i))_{i\geq 0}$ be the Fibonacci-Thue-Morse sequence; in other words, $t(i)$ is the number of 1s in the Fibonacci representation of i modulo 2. For a constant $c \geq 0$, the number of states in the minimal DFAO generating $(t(i+c))_{i\geq 0}$ is $O(c)$.

Immediately after the proof they write "We suspect that the number of states in a minimal automaton generating $(t(i+c))_{i\geq 0}$ has a linear lower bound too. The following is left as an open problem," and state

> **Problem 1.** Prove the number of states in a minimal automaton generating $(t(i+c))_{i\geq 0}$ is $\Theta(c)$.

(Numbers are those printed in the arXiv PDF of [1], v1, where theorems, lemmas and the other numbered environments share one counter and problems have their own; we verified them on the e-print compiled from the authors' source.) The paper prints no value of this quantity for any c; nor does the first author's thesis [3], whose Theorem 48 and Problem 3 (p. 58) repeat the two statements above word for word, nor the base-k companion paper [2], which for the binary Thue–Morse sequence proves an upper bound of $\frac{10}{3}c$ states (its Theorem 19) and a lower bound of more than $c^{0.694}$ states for infinitely many c (its Theorem 20), and asks for a formula for the exact number of states (its Open Problem 2; the values are OEIS A382296 and A382298 [7]).

Write $\mathrm{sc}(c)$ for the number of states of a minimal automaton generating $(t(i+c))_{i\geq 0}$ in the model of [1], made precise in Definition 2.2. This paper determines $\mathrm{sc}(c)$ exactly on an initial segment.

Theorem 1.1 (= Theorem 5.1)**.** *For every $0 \leq c \leq 100$, $\mathrm{sc}(c) = L(c)$, where $L(c)$ is the value printed in Table 1. In particular $\mathrm{sc}(0) = 4$, $\mathrm{sc}(100) = 568$, every $L(c)$ is even, L is not monotone ($L(12) = 78 > 74 = L(13)$), and $5c < \mathrm{sc}(c) \leq 7c$ for $2 \leq c \leq 100$.*

The lower half of Theorem 1.1 is the open half of Problem 1, made exact for each c in the range; the upper half is the construction of [1, Theorem 13] minimised. The two halves are established differently and are the two contributions of the paper.

Lower bounds by certificates. A Myhill–Nerode argument adapted to the partial automata of [1] (Theorem 3.1) turns a finite list of input words, pairwise distinguished either by their last digit or by a common valid continuation on which the outputs differ, into a lower bound. For each $c \leq 100$ a list of exactly $L(c)$ such words was found by computer and its pairwise separation was checked in the proof assistant (Theorem 3.2).

Upper bounds by a verified construction. The automaton of [1, Theorem 13], specialised to t, has as states the length-$(c+1)$ windows of the sequence $u(i) = (t(i), f(i))$, and its transition is a substitution rule on windows. We prove that rule for every window length and every position (Theorem 4.2), and prove that any transition table passing a finite consistency check is a correct automaton in the strict sense of [1] — correct output on every valid input, no state at all on any invalid one (Theorem 4.3). The minimised tables for $c \leq 100$ pass the check. Lean's type checker then forces the lower and upper halves to name the same integer.

What is not proved. Problem 1 itself — a linear lower bound for *all* c — is not settled here, and no statement about $\mathrm{sc}(c)$ for $c > 100$ is a theorem of this paper. Section 6 records, as computations outside the formal development, the exact values to $c = 300$ (with $5.16\,c < \mathrm{sc}(c) \leq 7c$ on $2 \leq c \leq 300$) and an empirical closed form for the size of the unminimised construction.

Attribution. The mathematics of the construction in Section 4 is that of [1, Theorems 13 and 14]; what is added is a proof, for every shift, that it satisfies the "no state on an invalid input" convention which [1] assumes, and the finite check that lets a table stand in for the construction. The count of four states for t is Shallit's [4]; its minimality (Theorem 4.4) is not stated there and is trivial. Zeckendorf uniqueness is classical [6, 5]. The transition rule of the construction is, under a relabelling, the morphism $1 \to 12$, $2 \to 4$, $3 \to 1$, $4 \to 43$ that Dekking recorded for A095076 [7]; that it is the transition of the automaton is what Theorem 4.2 proves.

Organisation. Section 2 fixes the model, including the convention that changes the answer at $c = 0$ from 2 to 4. Section 3 proves the lower bounds, Section 4 the construction and the upper bounds, Section 5 the exact values and their controls. Section 6 contains the computations that are not theorems, Section 7 describes the verification, and Section 8 lists what remains open. Appendix A prints the formal statements.

2. Preliminaries

2.1. **Words and Zeckendorf representations.** A *word* is a finite sequence over $\{0,1\}$, written most significant digit first; ε is the empty word, $|w|$ the length, $|w|_1$ the number of ones, and for $w \neq \varepsilon$ we write $\ell(w)$ for its last digit, with $\ell(\varepsilon) = 0$. A word is *valid* if it has no factor 11; a prefix of a valid word is valid, and so is $0^k w$ whenever w is. The value of $w = e_{m+1} \cdots e_2$ is $[w] = \sum_{k=2}^{m+1} e_k F_k$, and appending a digit obeys the identities

$$[xa] = [x0] + a, \qquad [xa0] = [x] + [x0] + 2a \qquad (a \in \{0,1\}), \tag{1}$$

the second being [1, Lemma 2(a)] with a digit in the middle. The greedy algorithm produces, for each $i \in \mathbb{N}$, a valid word (i) without leading zero and with $[(i)] = i$; we put $t(i) = |(i)|_1 \bmod 2$ and $f(i) = \ell((i))$. Every valid word of length m has value below F_{m+2}, and this gives uniqueness in the form we need.

Proposition 2.1 (Zeckendorf uniqueness)**.** *Let w be a valid word. Then $t([w]) = |w|_1 \bmod 2$ and $f([w]) = \ell(w)$. More precisely, two valid words of the same length and the same value are equal, so a valid word is the greedy word of its value at its own length, and $[x0]$ depends only on $[x]$ for valid x.*

Proof. Induction on the length after padding both words with leading zeros to a common length: if the leading digits differ, the value bound $[w] < F_{|w|+2}$ (and $< F_{|w|+1}$ when w starts with 0) makes the values differ. The first sentence follows by comparing w with $([w])$. □

We write $\sigma(m) = [(m)0]$; by Proposition 2.1, $[w0] = \sigma([w])$ and $[w1] = \sigma([w]) + 1$ for every valid w. The *interior sequence* of t is $u(i) = (t(i), f(i)) \in \{0,1\}^2$; by Theorem 4.4 below it is the interior sequence in the sense of [1, §2.1], the sequence of states of the minimal automaton for t.

2.2. **The automaton model.**

Definition 2.2. A *Fibonacci-DFAO* with n states is a triple (δ, τ, q_0) with $q_0 \in Q = \{0, \dots, n-1\}$, output $\tau\colon Q \to \{0,1\}$ and a *partial* transition $\delta\colon Q \times \{0,1\} \rightharpoonup Q$, extended to words by $\delta(q, \varepsilon) = q$ and $\delta(q, aw) = \delta(\delta(q,a), w)$, undefined as soon as one step is. It *generates* $g\colon \mathbb{N} \to \{0,1\}$ if

(G1) for every valid word w, $\delta(q_0, w)$ is defined and $\tau(\delta(q_0, w)) = g([w])$;
(G2) for every invalid word w, $\delta(q_0, w)$ is undefined.

The *state complexity* $\mathrm{sc}(g)$ is the least n for which an n-state Fibonacci-DFAO generates g, and $\mathrm{sc}(c) := \mathrm{sc}\big(i \mapsto t(i+c)\big)$.

The two clauses are the two conventions of [1]. (G1) contains the leading-zero convention ("the automata we discuss give the same behavior (accept/reject or output) no matter how many leading zeros are appended to the front of the input", [1, §2.1]), since $0^k w$ is valid whenever w is. (G2) is the sentence "We assume that such a DFAO does not reach any state on an input containing an invalid Fibonacci representation" of [1, §2.1], restated in [1, §4] as "if there is a state q reachable on 1 from some other state, then $\delta_h(q, 1)$ is undefined." In the formal development $\mathrm{sc}(g) = m$ is the statement that m is the least element of the set of sizes of automata generating g (Appendix A), so every exact value below is a pair: an automaton, and a proof that no smaller one exists.

Clause (G2) is not decoration; it is what makes $\mathrm{sc}(0) = 4$ rather than 2.

Proposition 2.3 (the convention is load-bearing)**.** *Let P be the two-state automaton that records the parity of the ones read so far, with both transitions defined at both states and output the parity. Then P satisfies (G1) for $g = t$ — it reaches a state with output $t([w])$ on every valid word w — but reaches a state on the invalid word* 11*, so it is not a Fibonacci-DFAO generating t. Moreover the words* 101 *and ε satisfy $t([101z]) = t([z])$ for every z such that both $101z$ and z are valid: no common continuation distinguishes them, only their last digits do.*

Proof. The state of P after w is $|w|_1 \bmod 2$, which is $t([w])$ for valid w by Proposition 2.1; 11 is invalid and P reads it. For the second claim, $|101z|_1 = |z|_1 + 2$. □

3. Lower bounds

Theorem 3.1 (Myhill–Nerode for partial Fibonacci-DFAOs)**.** *Let A be a Fibonacci-DFAO with n states generating g, and let $w_1, \dots, w_k$ be valid words such that for every $i < j$ either $\ell(w_i) \neq \ell(w_j)$, or there is a word z with $w_i z$ and $w_j z$ both valid and $g([w_i z]) \neq g([w_j z])$. Then $k \leq n$.*

Proof. By (G1) each w_i reaches a state q_i; we show the q_i are distinct. If a common valid continuation z separates w_i and w_j, the states $\delta(q_i, z)$ and $\delta(q_j, z)$ carry different outputs, so $q_i \neq q_j$. If $\ell(w_i) = 0$ and $\ell(w_j) = 1$, then $w_i 1$ is valid, so $\delta(q_i, 1)$ is defined by (G1), while $w_j 1$ contains 11, so $\delta(q_j, 1)$ is undefined by (G2); hence $q_i \neq q_j$. □

The second case is the clause that the usual Myhill–Nerode statement lacks, and by Proposition 2.3 it cannot be dropped. Note that only *commonly* valid continuations are used: on a partial automaton a suffix valid after w_i but not after w_j separates nothing.

A *certificate* for the shift c is a pair (W, Z) of lists of words; it is *accepted* if every word of W is valid and, for every two words $x \neq y$ of W (as list positions), either $\ell(x) \neq \ell(y)$ or some $z \in Z$ has xz, yz valid and $t([xz] + c) \neq t([yz] + c)$. Acceptance is a Boolean function of (c, W, Z); it is evaluated by computing, once per word, the vector of outputs $t([xz] + c)$ over $z \in Z$ (with a marker where xz is invalid) and comparing vectors pairwise. By Theorem 3.1, an accepted certificate with $|W| = L$ gives $\mathrm{sc}(c) \geq L$.

Theorem 3.2 (lower bounds)**.** *For every $0 \leq c \leq 100$ there is an accepted certificate (W_c, Z_c) with $|W_c| = L(c)$, the value in Table 1; hence $\mathrm{sc}(c) \geq L(c)$.*

TABLE 1. $L(c) = \mathrm{sc}(c)$ for $0 \le c \le 100$; the entry in row r and column j is $L(r + j)$.

c	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9
0	4	8	12	20	28	28	38	44	48	56
10	64	70	78	74	86	94	96	108	108	116
20	120	120	132	138	146	156	152	166	174	170
30	186	188	188	200	186	208	214	210	234	220
40	240	252	236	264	258	266	280	264	286	294
50	294	304	292	306	312	298	322	330	330	348
60	332	360	370	356	378	376	388	400	374	408
70	414	404	428	410	432	444	422	454	452	460
80	476	464	476	482	460	496	486	490	500	472
90	508	520	512	538	510	542	562	528	576	564
100	568									

Proof. The certificates were produced as follows. For each c the automaton of Section 4 was built, minimised by Moore refinement (undefined transitions treated as a dead class), and one shortest word reaching each class was taken as W_c; suffixes were then collected greedily from the valid words of length at most K, in order of length, one for each pair of words of W_c with equal last digit not yet separated, with K starting at 8 and increased until every pair was covered. The resulting lists have 29,256 words in all (length at most 16) and 1,480 suffixes (at most 25 per shift, length at most 11); for $c = 3$, for instance, $W_3 = (\varepsilon, 1, 10, 100, 101, 1000, \dots, 10010101)$ and $Z_3 = (\varepsilon, 0, 0000, 1)$. Acceptance of each of the 101 certificates, and the count $|W_c| = L(c)$, was then evaluated inside the proof assistant by compiled evaluation of the Boolean function above (Section 7); the lower bound follows from Theorem 3.1. Nothing in the proof depends on how the certificates were found. □

4. The construction of Moradi, Rampersad and Shallit, verified

4.1. **The successor and the child rule.** Reading one more digit sends a valid word w to $w0$ or $w1$, hence its value $m = [w]$ to $\sigma(m)$ or $\sigma(m) + 1$, the latter only when $\ell(w) = f(m) = 0$. We call these the *children* of m. The construction rests on how u behaves on children.

Proposition 4.1 (successor identities)**.** *For every $m \in \mathbb{N}$:*

(1) *$t(\sigma(m)) = t(m)$ and $f(\sigma(m)) = 0$; if $f(m) = 0$ then $t(\sigma(m) + 1) = 1 - t(m)$ and $f(\sigma(m) + 1) = 1$;*
(2) *$\sigma(m + 1) + f(m) = \sigma(m) + 2$.*

Proof. (1) is Proposition 2.1 read on the words $(m)0$ and $(m)1$. For (2) write words least significant digit first, so that appending a digit becomes prepending a letter, and define the successor ν by $\nu(\varepsilon) = 1$, $\nu(1r) = 0\,\nu(r)$, $\nu(0r) = 0\,\nu(r)$ if r starts with 1 and $\nu(0r) = 1r$ otherwise. With $v(r)$ the value of r and $s(r)$ the value of r with a 0 appended at the bottom, (1) becomes $v(ar) = s(r) + a$, $s(ar) = v(r) + s(r) + 2a$, and one simultaneous induction gives, for valid r: $\nu(r)$ is valid, $v(\nu(r)) = v(r) + 1$ and $s(\nu(r)) + \ell = s(r) + 2$ where ℓ is the lowest digit of r. Iterating ν from ε enumerates the valid words of value $0, 1, 2, \dots$, and $s(r) = \sigma(v(r))$ by Proposition 2.1. □

For $K \ge 0$ and $m \in \mathbb{N}$ let $W_K(m) = u(m)u(m + 1) \cdots u(m + K - 1)$ be the window of length K of u at m. Let E be the substitution on the alphabet $\{0, 1\}^2$ defined by

$$E(p, 1) = (p, 0), \qquad E(p, 0) = (p, 0)\,(1 - p, 1),$$

extended to words by concatenation, and let pref_K denote the prefix of length K. Under $A = (0, 0)$, $B = (0, 1)$, $C = (1, 0)$, $D = (1, 1)$ the substitution E is $A \to AD$, $B \to A$, $C \to CB$, $D \to C$, which is Dekking's morphism $1 \to 12$, $2 \to 4$, $3 \to 1$, $4 \to 43$ for A095076 [7] under $1 = A$, $2 = D$, $3 = B$, $4 = C$.

Theorem 4.2 (the window rule)**.** *For all $K \ge 0$ and $m \in \mathbb{N}$,*

$$W_K(\sigma(m)) = \mathrm{pref}_K\big(E(W_K(m))\big),$$

and if $f(m) = 0$ then $W_K(\sigma(m) + 1)$ is the prefix of length K of $E(W_K(m))$ with its first letter removed.

Proof. Induction on K, generalising m. Split on $f(m)$: if $f(m) = 1$, then $\sigma(m+1) = \sigma(m) + 1$ by Proposition 4.1(2), $E(u(m)) = u(\sigma(m))$ by (1), and the induction hypothesis at $m+1$ supplies the rest of the window; if $f(m) = 0$, then $\sigma(m+1) = \sigma(m) + 2$, $E(u(m)) = u(\sigma(m))\,u(\sigma(m)+1)$ by (1), and the hypothesis at $m+1$ (truncated by one letter) supplies the rest. The second statement is the first one at $m+1$, shifted by the two-letter block $E(u(m))$. □

Theorem 4.2 is the transition function of [1, Theorem 13] for $h = t$: the state reached on a valid w is $W_{c+1}([w])$, reading 0 applies E and keeps the first $c+1$ letters, reading 1 applies E, drops one letter and keeps $c+1$, and is allowed only when the first letter of the window has $f = 0$, i.e. when $\ell(w) = 0$. The output is the first coordinate of the last letter, $t([w] + c)$.

4.2. **From a checked table to an automaton.** A *configuration* for the shift c consists of integers $M, N \geq 1$, arrays $T_0, T_1 \in \{0, \dots, M-1\}^M$ and a class map $\kappa \in \{0, \dots, N-1\}^M$; the intended reading is that $0, \dots, M-1$ index the windows reachable from $W_{c+1}(0)$, T_a is the a-transition on windows, and κ maps windows onto the N states of a quotient automaton. The windows themselves are not part of the data: they are recomputed from $W_{c+1}(0)$ by walking T_0, T_1 in index order and applying the rule of Theorem 4.2, and every recomputed entry is then checked. The configuration is *accepted* if window 0 is $W_{c+1}(0)$ and, for every index i with representative j of its class (the largest index mapped to $\kappa(i)$): (a) the window at $T_0(i)$ is $\operatorname{pref}_{c+1} E$ of the window at i, and if the first letter of that window has $f = 0$, the window at $T_1(i)$ is the drop-one version; (b) $\kappa(T_0(j)) = \kappa(T_0(i))$ and, in the $f = 0$ case, $\kappa(T_1(j)) = \kappa(T_1(i))$; (c) windows i and j agree on the f-coordinate of their first letter and on the t-coordinate of their last.

Theorem 4.3 (a checked table is an automaton)**.** *If a configuration for the shift c with N states is accepted, then the automaton with states $\{0, \dots, N-1\}$, initial state $\kappa(0)$, transitions read off the class representatives ($\delta(q, 0) = \kappa(T_0(j))$, $\delta(q, 1) = \kappa(T_1(j))$ if the first letter of window j has $f = 0$ and undefined otherwise, j the representative of q) and output the t-coordinate of the last letter of the representative's window, is a Fibonacci-DFAO generating $(t(i + c))_{i \geq 0}$. Consequently* $\operatorname{sc}(c) \leq N$.

Proof. Induction on the word read, from the right: after a valid w the automaton is in class $\kappa(i)$ for some index i whose window is $W_{c+1}([w])$. Reading 0 keeps the invariant by (a), (b) and Theorem 4.2 with $[w0] = \sigma([w])$; reading 1 after a valid w with $\ell(w) = 0$ likewise, with $[w1] = \sigma([w]) + 1$, using (c) to know that the representative's window also has $f = 0$ in front, so the transition is defined. The output is right by (c) and the fact that the last letter of $W_{c+1}([w])$ is $u([w] + c)$. This is (G1). For (G2), an invalid word has a shortest invalid prefix $w1$ with w valid and $\ell(w) = 1$; then the first letter of the window of w has $f = f([w]) = \ell(w) = 1$ by Proposition 2.1, so by (c) the representative's window does too and the 1-transition is undefined. □

The check is not vacuous: at $c = 3$, corrupting one entry of T_0, merging the last class into the first (a 19-state claim, which must fail since $\operatorname{sc}(3) = 20$), reading the same tables at shift 4, or setting $M = 0$ each makes acceptance fail, and the automaton built from the accepted $c = 3$ table, run in the proof assistant on (i) for every $i < 60$, outputs $t(i+3)$ each time and reaches no state on 11 (Appendix A).

4.3. **The shift $c = 0$.** Shallit [4] gives a Fibonacci-DFAO for t (his Figure 1, states 0/0, 1/1, 2/1, 3/0 in the name/output notation) which, writing its states as $(p, b) \in \{0, 1\}^2$ with p the parity of the ones read and b the last digit read ($0/0 = (0,0)$, $1/1 = (1,1)$, $2/1 = (1,0)$, $3/0 = (0,1)$), has initial state $(0,0)$, $\delta((p,b),0) = (p,0)$, $\delta((p,0),1) = (1-p,1)$, $\delta((p,1),1)$ undefined and output p; he writes "it is easy to see that this DFAO has 4 states." It is the automaton to which the proof of [1, Theorem 14] refers.

Theorem 4.4. *Shallit's four-state automaton is a Fibonacci-DFAO generating t in the sense of Definition 2.2, and* $\operatorname{sc}(0) = 4$*: no Fibonacci-DFAO with fewer than four states generates t.*

Proof. On a valid w the automaton reaches $(|w|_1 \bmod 2, \ell(w))$ (induction from the right), whose output is $t([w])$ by Proposition 2.1; on an invalid word it stops at the first 11. The certificate $W_0 = (\varepsilon, 1, 10, 101)$, $Z_0 = (\varepsilon)$ is accepted: $\varepsilon, 10$ end in 0 and $1, 101$ in 1, and $t(0) \neq t([10]) = t(2)$, $t(1) \neq t([101]) = t(4)$. □

Thus the published count is the minimum, and (G2) is what makes it so (Proposition 2.3).

5. Exact values

Theorem 5.1 (exact values)**.** *For every $0 \le c \le 100$, $\mathrm{sc}(c) = L(c)$ with $L(c)$ as in Table 1.*

Proof. The lower bound is Theorem 3.2. For the upper bound, the automaton of Section 4 was built for each c by breadth-first search from $W_{c+1}(0)$ (this yields M windows), minimised by Moore refinement into N classes, and the resulting (T_0, T_1, κ) was written down as a configuration; acceptance was evaluated in the proof assistant by compiled evaluation of the check of Theorem 4.3. In each case $N = L(c)$: the formal statement $\mathrm{sc}(c) = L(c)$ is literally the pair (accepted configuration with N states, accepted certificate with $L(c)$ words), and it type-checks only if N and $L(c)$ are the same numeral. At $c = 0$ the four states are the four windows of length one, $M = N = 4$; at $c = 100$, $M = 690$ and $N = 568$. □

Corollary 5.2. *For $2 \le c \le 100$, $5c < \mathrm{sc}(c) \le 7c$, with equality on the right only at $c = 4$ and the least ratio $\mathrm{sc}(89)/89 = 5.30\ldots$; every $\mathrm{sc}(c)$, $c \le 100$, is even; and $\mathrm{sc}(c+1) < \mathrm{sc}(c)$ for 27 values of $c < 100$, the first being $c = 12$.*

Proof. Read off Table 1. □

Proposition 5.3 (controls)**.** *The following are verified alongside the theorems, to guard against a misplaced definition. (i) $t(0), \ldots, t(19) = 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1$, the twenty terms printed in* [4] *and the first twenty of A095076, and $f(0), \ldots, f(7) = 0, 1, 0, 0, 1, 0, 1, 0$, computed from the same Zeckendorf code path. (ii) $t(0) \ne t(1)$; no three-state Fibonacci-DFAO generates t; neither 3 nor 5 is $\mathrm{sc}(0)$. (iii) The certificate for $c = 3$ is rejected with an empty suffix list, with a repeated word, and when evaluated at shift 4. (iv) At $c = 7$, 43 states do not suffice and 45 is not the least sufficient number; at $c = 100$ the same holds for 567 and 569.*

6. Outside the formal development

The statements of this section are computations, carried out in exact integer arithmetic, and are not theorems of this paper.

Remark 6.1 (values to $c = 300$). Building and minimising the construction of Section 4 for every $c \le 300$ takes under a minute. The resulting $\mathrm{sc}(c)$ agree with Table 1 for $c \le 100$ (where they are proved); they were further validated by running each automaton on (i) for all $i < 1500$ and $c \le 40$, and by an independent Myhill–Nerode count straight from the sequence (all valid words of length at most 12, suffixes of length at most 9, the separation relation of Theorem 3.1), which returns the same number of classes for every $c \le 12$ with no unseparated pair. On $0 \le c \le 300$ the value is always even; $\mathrm{sc}(c)/c$ lies in $[5.1667, 7]$ for $2 \le c \le 300$, the minimum at $c = 288$ and the maximum at $c = 4$; $\mathrm{sc}(298), \mathrm{sc}(299), \mathrm{sc}(300) = 1626, 1652, 1672$; the first differences take 77 distinct values between -96 and 108. No closed form is in sight, and neither the sequence $\mathrm{sc}(c)$ nor the sequence $M(c)$ of the next remark is in the OEIS (searches on the first 15 and 16 terms, 3 September 2026, with a positive control).

Remark 6.2 (the size of the construction). By Theorem 4.2 the states of the unminimised construction reachable from $W_{c+1}(0)$ are exactly the windows $W_{c+1}(m)$, $m \ge 0$, so their number $M(c)$ is the factor complexity $\rho_u(c+1)$ of u: $\rho_u(1), \rho_u(2), \cdots = 4, 10, 16, 22, 28, 34, 42, 48, 54, 62, 70, 76, \ldots$, and $\rho_u(101) = 690$. For every $n \le 201$ the first differences obey

$$\rho_u(n+1) - \rho_u(n) = \begin{cases} 8 & \text{if } F_k < n \le F_k + F_{k-3} \text{ for some } k \ge 5, \\ 6 & \text{otherwise,} \end{cases}$$

the runs of 8 being $[6, 6]$, $[9, 10]$, $[14, 16]$, $[22, 26]$, $[35, 42]$, $[56, 68]$, $[90, 110]$, $[145, 178]$, of lengths $F_2, F_3, F_4, \ldots$; equivalently $\rho_u(n) = 6n - 2 + 2\,\#\{1 \le m < n : m \text{ in one of these runs}\}$. If the rule persists, $\rho_u(n)/n \to 6 + 2\varphi^{-2} = 6.7639\ldots$, φ the golden ratio; on $2 \le n \le 201$ the ratio lies in $[5, 6.961]$. This is the quantity that [1, Theorem 14] bounds by $O(c)$ through the linear factor complexity of a Fibonacci-automatic sequence [1, Lemma 1], here with an explicit slope, empirically. The analogous statement for the two-letter sequence t itself is Dekking's conjecture, which Shallit [4] states as his Conjecture 1 — the first differences of the factor complexity of t are $(1, 2, 4, 6, 10) \prod_{i \ge 2} 6^{F_i + (-1)^i} 8^{F_i}$, the product denoting concatenation and the exponents repetition — and proves, his Theorem 2 giving the differences run by run. It is a different pattern from ours (it has a 10, and its runs of 6 have length $F_i \pm 1$); our implementation reproduces the first 21 of those

differences, $1, 2, 4, 6, 10, 6, 6, 8, 6, 8, 8, 6, 6, 6, 6, 8, 8, 8, 6, 6, 6$, as printed there. Minimisation merges $\rho_u(c+1) - \mathrm{sc}(c)$ windows; this count starts 0, 2, 4, 2, 0, 6, 4, 4, 6, 6, 6, 6, 4, 14, 10, 10, 16, 10, 16, 14, 16 for $c = 0, 1, 2, \ldots$, and vanishes only at $c = 0$ and $c = 4$ within $c \le 300$.

7. Verification

Every theorem, proposition and corollary of Sections 2–5 is proved in Lean 4 with Mathlib, in fourteen modules of 4,146 lines, free of `sorry`; the formal statements are printed in Appendix A. The reasoning layer — Zeckendorf uniqueness, the Myhill–Nerode bound, the soundness of certificates, the successor identities, the window rule for all K and m, the correctness of a checked table, and Shallit's automaton — depends only on the axioms `propext`, `Classical.choice` and `Quot.sound` (the successor identity `nxtR_spec` on `propext` and `Quot.sound` alone). What is delegated to compiled evaluation (`native_decide`) is exactly the finite checks: the 101 certificate acceptances of Theorem 3.2 (49 in one module, 23 s of wall time, and 52 in another, 125 s), the 101 configuration acceptances of Theorem 5.1 (49 and 52, 6 s and 15 s), and the twelve controls of Section 5 and Theorem 4.3; peak memory stayed below 2.3 GB. Lean records each such evaluation as its own axiom, so each exact value $\mathrm{sc}(c) = L(c)$ depends on precisely two of them, one per half, and on nothing else beyond the three standard axioms; `#print axioms` was run on every declaration named in Appendix A and the outcome is as stated. The certificates and tables are carried as strings (342 kB and 390 kB) and parsed inside the evaluation. The computations of Section 6 were done in Python with exact integers; the Lean development does not depend on them, and the certificate and table generators are not trusted — the proof assistant re-derives every window and every separation. Repository reference to be added at submission.

8. Open questions

(1) *Problem 1 of* [1]. A linear lower bound for all c is untouched here. Table 1 and Remark 6.1 say that $\mathrm{sc}(c) \ge 5c$ holds for $2 \le c \le 300$ with slack at least $0.16\,c$; for the binary Thue–Morse analogue the companion paper proves an upper bound of $\frac{10}{3}c$ but, from below, only that the minimal automaton has more than $c^{0.694}$ states for infinitely many c [2, Theorems 19 and 20], so a linear lower bound may be genuinely hard.

(2) *A general upper bound in the kernel.* Theorem 4.3 reduces $\mathrm{sc}(c) \le N$ for all c to a proved bound on $\rho_u(c+1)$; the rule of Remark 6.2 would give $\mathrm{sc}(c) \le \rho_u(c+1) \le 7c + 7$, say, but proving it is the u-analogue of Dekking's conjecture, which for t was settled by a decision procedure [4] rather than by hand.

(3) *Beyond $c = 100$.* The upper half is cheap (about 2.4 s per shift at $c \approx 300$); the lower half is what costs, its certificate check growing like $\sum_c L(c)^2 |Z_c|$.

(4) *A formula for* $\mathrm{sc}(c)$. Unlike the shifts of the Fibonacci word, for which [1] announces an $O(\log c)$ bound proved in a forthcoming paper of Moradi, Popoli, Shallit and Vukusic, nothing in the data suggests a closed form for $\mathrm{sc}(c)$ or for the merge count of Remark 6.2.

Appendix A. The formal statements

The development is in the namespace `LeanProblemSpec.FtmShiftStates`; `Word` is `List Bool` read most significant digit first, `wvalid` is validity, `fval` the value $[\cdot]$, `ones` the number of ones, `lastD` the last digit, `zk` the greedy representation, and `ftm`, `fibw` are t and f. The statements below are quoted verbatim (proofs omitted); the declarations named in the margins of the text are these.

The model (Definition 2.2).

```
def ftm (n : ℕ) : Bool := decide (ones (zk n) % 2 = 1)
def fibw (n : ℕ) : Bool := lastD (zk n)
def shifted (c : ℕ) : ℕ → Bool := fun i => ftm (i + c)
structure DFAO (n : ℕ) where
  step : Fin n → Bool → Option (Fin n)
  out : Fin n → Bool
  init : Fin n
def Computes {n : ℕ} (A : DFAO n) (g : ℕ → Bool) : Prop :=
  (∀ w : Word, wvalid w = true → ∃ q, A.run A.init w = some q ∧ A.out q = g (fval w)) ∧
  (∀ w : Word, wvalid w = false → A.run A.init w = none)
def SizeSet (g : ℕ → Bool) : Set ℕ := {n | ∃ A : DFAO n, Computes A g}
def SC (g : ℕ → Bool) (m : ℕ) : Prop := IsLeast (SizeSet g) m
```

Proposition 2.1.

```
theorem ftm_of_word (w : Word) (hw : wvalid w = true) :
    ftm (fval w) = decide (ones w % 2 = 1)
theorem fibw_of_word (w : Word) (hw : wvalid w = true) : fibw (fval w) = lastD w
```

Proposition 2.3.

```
def A2' : DFAO 2 where
  step q a := some (if a then (if q.val % 2 = 1 then 0 else 1) else q)
  out q := decide (q.val % 2 = 1)
  init := 0
theorem parity_computes_valid (w : Word) (hw : wvalid w = true) :
    ∃ q, A2'.run A2'.init w = some q ∧ A2'.out q = shifted 0 (fval w)
theorem parity_reaches_on_11 :
    wvalid [true, true] = false ∧ A2'.run A2'.init [true, true] ≠ none
theorem A2'_not_computes : ¬ Computes A2' (shifted 0)
theorem lastD_is_not_a_suffix_separation (z : Word)
    (h1 : wvalid ([true, false, true] ++ z) = true) (h2 : wvalid ([] ++ z) = true) :
    shifted 0 (fval ([true, false, true] ++ z)) = shifted 0 (fval ([] ++ z))
theorem lastD_separates_them : lastD [true, false, true] ≠ lastD ([] : Word)
```

Theorem 3.1 and the certificate (`okCert` *is acceptance).*

```
theorem length_le_of_pairwise_sep {n : ℕ} (A : DFAO n) (g : ℕ → Bool)
    (hA : Computes A g) (ws : List Word)
    (hval : ∀ w ∈ ws, wvalid w = true)
    (hsep : ws.Pairwise (fun x y => lastD x ≠ lastD y ∨ ∃ z, wvalid (x ++ z) = true ∧
        wvalid (y ++ z) = true ∧ g (fval (x ++ z)) ≠ g (fval (y ++ z)))) :
    ws.length ≤ n
def okCert (g : ℕ → Bool) (Z : List Word) (ws : List Word) : Bool :=
  ws.all wvalid && pairwiseB sepP (sigList g Z ws)
theorem mem_lowerBounds_of_okCert (g : ℕ → Bool) (Z : List Word) (ws : List Word)
    (h : okCert g Z ws = true) : ws.length ∈ lowerBounds (SizeSet g)
def cert (c L : Nat) : Bool := ((Wc c).length == L) && okCert (shifted c) (Zc c) (Wc c)
theorem lower_of_cert {c L : Nat} (h : cert c L = true) :
    (L : Nat) ∈ lowerBounds (SizeSet (shifted c))
```

Theorem 3.2: one declaration `lower_c` *per shift (*`Wc`*,* `Zc` *are the parsed lists;* `lower_of_cert2`*,* `cert2` *are the same for* $49 \le c \le 100$*). The first and the last:*

```
theorem lower_0 : (4 : ℕ) ∈ lowerBounds (SizeSet (shifted 0)) :=
  lower_of_cert (c := 0) (L := 4) (by native_decide)
theorem lower_100 : (568 : ℕ) ∈ lowerBounds (SizeSet (shifted 100)) :=
  lower_of_cert2 (c := 100) (L := 568) (by native_decide)
```

Proposition 4.1 (`vR`*,* `uR` *are* v*,* s*;* `hd` *the lowest digit;* `upN` *is* σ*).*

```
def nxtR : List Bool → List Bool
  | [] => [true]
  | true :: r => false :: nxtR r
  | false :: r => if hd r then false :: nxtR r else true :: r
theorem nxtR_spec : ∀ r : List Bool, wvalid r = true →
    vR (nxtR r) = vR r + 1 ∧
    uR (nxtR r) + (if hd r then 1 else 0) = uR r + 2 ∧
    wvalid (nxtR r) = true
def upN (m : ℕ) : ℕ := fval (zk m ++ [false])
theorem upN_succ (m : ℕ) : upN (m + 1) + (if fibw m then 1 else 0) = upN m + 2
```

Theorem 4.2 (`wins K m` *is* $W_K(m)$*,* `kidSig` *is* E*).*

```
def uL (m : ℕ) : Letter := (ftm m, fibw m)
def wins : ℕ → ℕ → List Letter
  | 0, _ => []
  | (K+1), m => uL m :: wins K (m+1)
def expand (x : Letter) : List Letter :=
  if x.2 then [(x.1, false)] else [(x.1, false), (!x.1, true)]
def kidSig (s : List Letter) : List Letter := s.flatMap expand
theorem wins_upN : ∀ (K m : ℕ), wins K (upN m) = (kidSig (wins K m)).take K
theorem wins_upN_succ : ∀ (K m : ℕ), fibw m = false →
    wins K (upN m + 1) = ((kidSig (wins K m)).drop 1).take K
```

Theorem 4.3 (`Cfg` *is a configuration,* `okCfg` *its acceptance,* `okAt` *the check at one index,* `mkDFAO` *the automaton).*

```
theorem computes_of_checks (D : Cfg) (S : Array (List Letter)) (R : Array ℕ)
    (hN : 0 < D.N) (hM : 0 < D.M) (hS0 : S[0]! = win D.c 0)
    (hall : ∀ i, i < D.M → okAt D S R i = true) :
    Computes (mkDFAO D hN S R) (shifted D.c)
theorem sizeSet_of_okCfg (D : Cfg) (h : okCfg D = true) : D.N ∈ SizeSet (shifted D.c)
theorem okCfg_rejects_bad_transition : okCfg cfgBadT0 = false
theorem okCfg_rejects_merge : okCfg cfgMerged = false
theorem okCfg_rejects_wrong_shift : okCfg cfgWrongShift = false
theorem okCfg_rejects_empty : okCfg cfgEmpty = false
theorem A3_dead_on_11 : A3.run A3.init [true, true] = none
```

Theorem 4.4 (`A0` is Shallit's automaton).

```
theorem A0_computes : Computes A0 (shifted 0)
theorem four_mem_sizeSet : (4 : ℕ) ∈ SizeSet (shifted 0)
theorem sc_zero : SC (shifted 0) 4
```

Theorem 5.1: one configuration `cfg_c`*, one acceptance* `okCfg_c` *and one value* `scExact_c` *per shift. For* $c = 0$ *in full, and the last:*

```
def cfg0 : Cfg := mkCfg 0 4
  "0,2,2,0"
  "1,0,3,0"
  "0,1,2,3"
theorem okCfg0 : okCfg cfg0 = true := by native_decide
theorem scExact_0 : SC (shifted 0) 4 :=
  ⟨sizeSet_of_okCfg cfg0 okCfg0, lower_0⟩
theorem scExact_100 : SC (shifted 100) 568 :=
  ⟨sizeSet_of_okCfg cfg100 okCfg100, lower_100⟩
```

Proposition 5.3.

```
theorem ftm_prefix :
    (List.range 20).map ftm =
      [false, true, true, true, false, true, false, false, true, false,
       false, false, true, true, false, false, false, true, false, true]
theorem fibw_prefix :
    (List.range 8).map fibw = [false, true, false, false, true, false, true, false]
theorem ftm_not_constant : ftm 0 ≠ ftm 1
theorem no_three_state : (3 : ℕ) ∉ SizeSet (shifted 0)
theorem not_sc_five : ¬ SC (shifted 0) 5
theorem not_sc_three : ¬ SC (shifted 0) 3
theorem cert_needs_suffixes : okCert (shifted 3) [] (Wc 3) = false
theorem cert_rejects_duplicate :
    okCert (shifted 3) (Zc 3) (Wc 3 ++ [(Wc 3).headI]) = false
theorem cert_rejects_wrong_shift : okCert (shifted 4) (Zc 3) (Wc 3) = false
theorem no_43_state_7 : (43 : ℕ) ∉ SizeSet (shifted 7)
theorem not_sc_45_at_7 : ¬ SC (shifted 7) 45
theorem no_567_state_100 : (567 : ℕ) ∉ SizeSet (shifted 100)
theorem not_sc_569_at_100 : ¬ SC (shifted 100) 569
theorem A3_runs_correctly :
    ((List.range 60).all fun i => ((A3.run A3.init (zk i)).map A3.out) == some (shifted 3 i))
      = true
```

References

[1] D. Moradi, N. Rampersad and J. Shallit, *Complexity of Linear Subsequences of Fibonacci-Automatic Sequences*, arXiv:2603.21645v1, 23 March 2026, cs.FL, cs.DM and math.CO. Numbered statements are cited as printed in the arXiv PDF (theorems, lemmas and the other numbered environments share one counter; problems have their own), checked against the e-print compiled from the authors' source: Lemma 1 (§2.1, p. 3), Lemma 2 (§2.2, p. 5), Theorem 13 (§4, p. 12), Theorem 14 and Problem 1 (§4, p. 13).

[2] D. Moradi, N. Rampersad and J. Shallit, *Complexity of Linear Subsequences of k-Automatic Sequences*, arXiv:2512.10017, first posted 10 December 2025; the statements cited here are numbered as in v5 (2 April 2026): Theorems 19 and 20 and Open Problem 2, all in its §4.2, p. 17.

[3] D. Moradi, *State Complexity of Linear Relations and Linear Subsequences of Automatic Sequences*, thesis for the degree of Master of Mathematics in Computer Science, University of Waterloo, 2026, `hdl.handle.net/10012/23044` (issued 23 April 2026); Theorem 48 and Problem 3 are on its p. 58.

[4] J. Shallit, *Subword complexity of the Fibonacci–Thue–Morse sequence: the proof of Dekking's conjecture*, Indagationes Mathematicae 32 (2021), no. 3, 729–735, doi:10.1016/j.indag.2021.03.004; arXiv:2010.10956. Conjecture 1, Theorem 2 and Figure 1 are numbered as in arXiv v3 (8 November 2020).

[5] E. Zeckendorf, *Représentation des nombres naturels par une somme de nombres de Fibonacci ou de nombres de Lucas*, Bull. Soc. Roy. Sci. Liège 41 (1972), 179–182.

[6] C. G. Lekkerkerker, *Voorstelling van natuurlijke getallen door een som van getallen van Fibonacci*, Simon Stevin 29 (1952), 190–195.

[7] OEIS Foundation Inc., *The On-Line Encyclopedia of Integer Sequences*, `oeis.org`: A095076 (parity of the number of ones in the Zeckendorf expansion, with M. Dekking's comment of 28 November 2019 giving the morphism $1 \to 12$, $2 \to 4$, $3 \to 1$, $4 \to 43$ and coding $1, 3 \mapsto 0$, $2, 4 \mapsto 1$, and his comment of 10 March 2020 giving the same morphism on the pair alphabet, $(0,0) \to (0,0)(1,1)$, $(0,1) \to (0,0)$, $(1,0) \to (1,0)(0,1)$, $(1,1) \to (1,0)$, with $(0,0), (0,1), (1,0), (1,1) \mapsto 1, 3, 4, 2$), A382296 and A382298 (states of the smallest DFAO computing the binary Thue–Morse sequence shifted by n, msd-first and lsd-first); consulted 3 September 2026.